H D C / D M A H A R D D I S K C O N T R O L L E R

T E C H N I C A L R E F E R E N C E M A N U A L

DISCLAIMER

No representations or warranties, express or implied, are made with respect to the contents hereof, including, but not limited to, the implied warranty of merchantability or fitness for a particular purpose. Further, Morrow Inc., reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision.

Morrow
600 McCormick St.
San Leandro, CA 94577

IMPORTANT WARRANTY INFORMATION

LIMITED WARRANTY

Morrow, Inc. warrants its products to be free from defects in workmanship and materials for the periods indicated below. This warranty is limited to the repair and replacement of parts only.

This warranty is void if, in the sole opinion of Morrow Inc., the product has been subject to abuse or misuse, or has been interconnected to other manufacturer's equipment for which compatibility has not been established in writing.

Circuit boards - Parts, including the printed circuit board, purchased as factory assemblies, are warranted for a period of ninety (90) days from the original invoice/purchase date.

Electro-mechanical peripherals - Peripheral equipment such as floppy or hard disk drives, etc., not manufactured by Morrow, Inc., are included in the limited warranty period of 90 days from the original invoice date when sold as part of a Morrow system.

Exception - Expendable items such as printer ribbons, software media, and printwheels are not covered by any warranty.

Software/Firmware - Morrow, Inc. makes no representations or warranties whatsoever with respect to software or firmware associated with its products and specifically disclaims any implied or expressed warranty of fitness for any particular purpose or compatibility with any hardware, operating system, or software/firmware. Morrow, Inc. reserves the right to alter or update any program, publication or manual without obligation to notify any person of such changes.

LIMITATION OF LIABILITY: THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MORROW, INC. BE LIABLE FOR CONSEQUENTIAL DAMAGES EVEN IF MORROW, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

WARRANTY RETURN PROCEDURE

Should a buyer experience a defect in either workmanship or materials during the warranty period, any Morrow Authorized Service Center will replace or repair the product at its expense only if the product is promptly returned to the dealer or Service Center with dated proof of purchase.

Should factory repair be necessary, the Service Center shall contact Morrow Customer Service for a Return Materials Authorization (RMA) number.

HDCDMA Hard Disk Controller

Technical Manual

Revision 1

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

The Hard Disk/Direct Memory Access (HDCDMA) Hard Disk Controller is a single board S-1ØØ bus subsystem. It can communicate with up to four 5 1/4 inch or four 8" hard disks. The HDCDMA will support a mixture of types of drives, but all must have the same transfer rate. Variable designs, capacities and sector formats are possible for each type of drive, allowing users to upgrade their systems with new drives as they become available.

The HDCDMA is an intelligent controller. The arrangement of data on the disk is not limited by a special purpose LSI controller chip. This allows system designers the freedom to specify sector sizes and interleave to fit the requirements of special applications.

The controller has its own 8X3ØØ bipolar microprocessor operating at 7.16 megahertz which is used to supervise data transfers between disk drives and memory without the intervention of the main CPU. This relieves the main CPU of time consuming processes which include head positioning, rotational delays and the usual byte-by-byte transfer of data from the controller to main memory. As a result, transfers are faster and more efficient. Moreover, the main CPU has more time for data processing, and thus, supports more users and/or tasks.

The main advantage of the HDCDMA controller over almost all the others is its glitch free direct memory access channel. This advanced "channel" concept allows the controller to communicate with S-1ØØ memory by "stealing" bus cycles from the main CPU, or using the bus in "burst mode" for ultra-fast data transfer. This idea of an intelligent I/O channel was first implemented by IBM on their famous 37Ø mainframes. Now, this powerful concept has been implemented on the S-1ØØ bus.

The channel has the full 24-bits of memory addressing as described in the proposed IEEE standard for the S-1ØØ bus. Also, a great deal of care has been taken in the design of the interface circuitry so that it conforms in every detail to this new standard and still allows the controller to work well with existing systems designed before the standardization effort was started.

The board has priority logic which allows it to contend with up to 15 other "temporary masters" which may also want to "steal" bus cycles from the main CPU (the permanent master).

The features associated with the intelligent channel on the controller make it especially desirable in multi-tasking and multi-user applications. In fact, many were tailored to enhance the performance of Morrow Designs new, powerful DECISION I multiprocessing IEEE 696/S-1ØØ machine. DMA style controllers are an integral part of advanced microcomputer systems which incorporate many of the concepts previously supported only on mainframe systems, such as the IBM 37Ø series.

The controller also provides a signal to the drives that is 1/16 the bit rate for compatibility with earlier versions of some drives. And special low voltage detection circuitry inhibits the WRITE ENABLE signal to the drives during power sequencing.

The operating system may be loaded by the HDCDMA by using a simple external bootstrapping program. This program in external ROM may be very simple because of the advanced design of the controller. This makes it possible to boot a hard disk directly, without the intermediate assistance of a floppy loaded operating system.

We do not know of any S-1ØØ bus hard disk controllers that come anywhere near the performance and versatility of the HD-DMA. For that matter, we here at Morrow Designs know of no other hard disk controller on **any** bus that can match the HDCDMA in price, power, performance and flexibility.

Good luck with this product! One of the purposes of this document is to detail how the HDCDMA Hard Disk Controller can improve the speed and performance of your system. If we've missed anything, please let us know.

2. THE CHANNEL CONCEPT

The IBM 370 mainframe was the first computer system to make use of the channel concept. Previously, an I/O controller, even one that could do direct memory accesses, was given commands and would report status by means of I/O ports. Usually, the commands were sent one at a time and status was reported through the status port.

The idea of storing both data and instructions in memory is what gave birth to the computers of today. IBM decided to extend this concept to DMA Controllers, which expanded the controllers capabilities in much the same way that computers became more powerful by being programmable.

Channel controllers differ from the CPU in that their task is to transfer data when it is required. Therefore,the CPU must have some way of starting the channel when it has tasks for it to perform. Using a memory location is not reliable because the contents of solid state memory are unknown whenever the power is first turned on. Also, polling a memory location is inefficient. The solution is to use an I/O port to start execution.

Using an I/O port to start the channel controller does not void the main advantage of this design: using memory to store commands for the controller. The channel controller carries out this list of commands without the supervision of the CPU. At the end of the commands, the controller writes its completion status in memory, and waits for another start command from the CPU.

Obviously, a channel type of controller needs some kind of on-board intelligence. At the time IBM first built this kind of device, it was expensive both in terms of dollars and in circuit board real estate to implement this intelligence. Today, however, the situation is quite different. Microprocessors are inexpensive and take only a modest amount of space on a circuit board.

In theory, the only limitation to the power and flexibility of a channel driven controller is the size of memory local to the resident microprocessor. Since memory is getting cheaper and denser, it would seem that time will favor the channel approach to I/O controllers.

2.1. HDCDMA Channel Command Structure

Command execution for the HDCDMA consist of a fixed three part sequence: reading a link field that points to the next command, seeking and executing an operation code. Every time the controller is started it follows this sequence. It is possible for the command to be null, that is, the heads will not be stepped and no operation will be performed. The link field must always be present because it is used to find the command when the controller is started.

The operation of the controller parallels the structure of the commands in memory. When the controller is started, it reads the address in the link field of the last command executed. This is the address of the next command to be executed.

Next the drive number, step direction and number of steps are transferred from memory pointed to by the link field to the controller. If the number of steps is greater than zero, the channel releases the bus and begins seeking.

When the seeking operation is complete, the controller collects the next group of parameters from the command structure. These consist of the DMA address, a set of four arguments and the operation code. The controller then performs the intended operation, and writes a status code into the main memory location following the operation code.

At this point the controller halts and awaits the next start command. A pointer internal to the controller retains the location of the link field. At the next start command, the contents of the link field are loaded by the controller as the beginning address of the new command.

The format of the channel commands is outlined below. Each of the three parts will be explained in detail in the sections that follow.

Table 2-1: Command Structure Format

	BYTE	NAME	FUNCTION
S	Ø	SELDRV	Select drive and step direction
E	1	STEP-L	Low byte of number of steps
E	2	STEP-H	High byte of number of steps
K			
	3	SEL-HD	Select drive, head, write pre-compensation and low-current
O	4	DMA-L	Low byte of DMA address
P	5	DMA-H	High byte of DMA address
E	6	DMA-E	Extended address byte of DMA
R	7	ARG-Ø	
A	8	ARG-1	Parameter list for
T	9	ARG-2	Operation codes
I	A	ARG-3	
O	B	OPCODE	Operation code
N	C	STATUS	Completion status is stored here
L	D	NEXT-L	Low byte of link address
I	E	NEXT-H	High byte of link address
N	F	NEXT-E	Extended address byte of link address
K			

2.1.1. Start and Reset Commands

These are the only two functions that make use of port addresses. The Reset command has the same effect as a POC* or a RESET* on the bus, which is to return the controller to its reset state. This command would be used if the controller became hung up trying to read an unformatted track.

The Start Command is used to initiate controller activity. The first Start Command after a reset fetches three bytes beginning at ØØØØ5ØH as a pointer to the beginning of a command structure. Stepping and command execution begin by reading the command structure at the address indicated by this pointer.

In subsequent commands, the controllers internal pointer is set to the address of the link field of the last command interpreted. This link field can be modified any time before the next start command is issued.

Care must be taken, however, to see to it that the link field always points to a valid command. Although it is unlikely, an undefined link field could point to a location in memory that contains the code for a Format Operation, resulting in a strangely formatted track and loss of data.

The port address used by the start command is 55H. Any OUTPUT to this port results in the initiation of controller activity. The value output to this port is ignored. The Start signal is named ATTN (for attention) on the HDCDMA schematics.

The port address for the Reset Command is 54H. Once again, any output instruction to this port has the immediate affect of resetting the controller regardless of its current activity. The value output to this port is ignored.

After a Reset Command, a 1Ø microseconds delay must occur before a Start Command may be issued.

2.1.2. Link field

The Link Field is a three byte address that points to the beginning of the next command structure to be executed. It is stored in the last three bytes of the previous command structure.

* An asterisk (*) following a signal name indicates that this signal uses negative logic. Thus, low (or Ø) is true and high (or 1) is false.

The controller maintains the address of the last command structure and reads the fourteenth, fifteenth and sixteenth bytes when it is restarted. There are only two ways of changing where the controller expects to find the Link Field: 1) Resetting the controller, or 2) starting the controller and using the Link Field to point to an address of its own structure. When the controller is Reset, it expects the first Link Field to begin at ØØØØ5ØH.

2.1.3. Stepping Commands

Stepping commands make up the first part of the channel command structure. After the controller has read the Link Field, it transfers the first three bytes of the command structure from main memory. If the number of steps is zero, then control is passed onto the operation part of the command structure. If the number of steps is non-zero, the controller releases the bus and commences stepping.

The first byte in the command structure is the SELDRV byte. It is constructed of two parts: The drive number and the stepping direction. The drive number is a binary number between Ø and 3, and is stored in the least significant two bits.

The stepping direction is stored in bit 4. When this bit is a Ø, the heads are stepped in, away from track Ø. If this bit is a 1, the heads are stepped out, toward track Ø. When stepping-out, if track Ø is encountered before the number of steps has been completed, stepping ends, and the controller begins the next part of the command. Since most drives do not detect over-stepping IN, care must be taken to prevent stepping past the inner-most cylinder.

Table 2-2: Format of BYTE Ø: DRVSEL

Bit #	7	6	5	4	3	2	1	Ø
BYTE Ø:	Ø	Ø	Ø	In/Out	Ø	Ø	Dr1	DrØ

Bits Ø and 1 (DrØ and Dr1) are used to select drives with a binary number between Ø and 3.

Bit 4 is used to select stepping DIRECTION: if it equals Ø, step in; if bit 4 = 1, step out.

Bits 2,3,5,6 and 7 are not used; may be the same as Byte 3 in drives with four heads or less.

The second and third bytes of the command structure are the low and high bytes of the number of steps. The number of steps can be any integer between Ø and 65535. If the number of steps is zero, no action is taken.

If the number of steps to be taken in the out direction is greater than the number of steps before track Ø, stepping halts at track Ø. Stepping in beyond the innermost cylinder should never be attempted as this may damage the drive.

NOTE: It is the operating system's responsibility never to step past the innermost track. Most drives do not produce a signal indicating this. Over-stepping-in may damage the drive. It may also result in positioning the heads over a non-formatted track, which requires that the controller be reset.

EXAMPLE: Suppose we wish to move the heads of drive 3 to cylinder 65H, (1Ø1 in decimal), and we are currently over cylinder 32H (5Ø). We need to step IN 33H steps on drive number three. The first three bytes of the command structure would be as follows:

```
BYTE Ø:  ØØØØ ØØ11  ;Bit 4 is Ø to step in
BYTE 1:  ØØ11 ØØ11  ;low byte contains 33H
BYTE 2:  ØØØØ ØØØØ  ;high byte contains Ø.
```

Because the HDCDMA is designed to handle a variety of drives, the time delay between steps and the head settle time are both under software control. These are explained in the section on the Load Constants Command.

2.1.4. Homing the Heads

The HDCDMA has built the Home function into the basic command structure, rather than having a separate operation for it. The Heads are homed by setting the Direction bit to a 1 for stepping OUT, and storing a value in the number of steps word that is greater than the maximum number of tracks for that drive. ØFFFH is a suggested value for ST5Ø6 drives.

When the heads reach track Ø, stepping ceases and the controller executes the next portion of the command structure. In drives with fast seeking capabilities, a step delay time sufficient to produce slow stepping should be specified using the Load Constants command. The on-board drive intelligence that controls ramping up or down for fast seeking is not aware of the location of the heads. Using fast seek for recalibration may result in the heads reaching track Ø while still accelerating and cause damage to the drive.

2.1.5. Operation Portion of the Command Structure

At the completion of stepping, the controller requests the bus, and then holds it while transferring the next nine bytes of the command into the controller's processor. The bus is then released if the controller is not ready to transfer data.

The nine bytes just read by the controller may be separated into four groups: the head select byte, the DMA address, the argument list and the operation code. It is important to set every byte to the correct value for each operation. The controller does not retain the DMA address, or any other argument, from the previous operation. The only values retained are those set by a Load Constants Command.

The first two groups, the head select byte and the DMA address, will be described first, as they are the same for all the commands. The argument lists will be explained in conjunction with the operation they are associated with.

At the end of any operation, successful or not, a status code is written into the byte following the operation code. This byte should be set to Ø initially, and can be monitored by the main CPU for completion of the disk operation in non-interrupt systems. A Ø in the status byte indicates the controller is busy.

2.1.6. Select Head Byte

The fourth byte of the command structure is the select head byte. This byte specifies both the drive number and the head to be used. It also contains information that is used when writing to the disk, write-precompensation and low current.

The least significant two bits of this byte are used to specify the drive number. The format used is identical to that in the SELDRV byte: the least significant two bits are set to a binary number between Ø and 3. The next three bits, 2, 3 and 4, are used to specify the head number, which is a binary number between Ø and 7. The ST5Ø6 uses negative logic to select heads. Thus, 7 selects head Ø, 1 selects 6, etc. This is drive dependent, so refer to the drive manual.

Bit 7 turns on write-precompensation. When this bit is true (set to a 1), the controller changes the timing of write pulses sent to the drive. This is done because as the cylinders get closer to the center of the disk, their circumferences become smaller and the packing density becomes greater.

Information is stored on the disk as transitions in the polarity of magnetization, and as these fields come closer together, they result in a time shift in the received data. Write-precompensation logic turns on the write gate slightly

earlier or later than normal to compensate for this interaction between fields. Write-precompensation is normally used beginning with cylinder 128 (80H) for the ST506. Please refer to the manufacturer's specificatons for other drives.

Bit 6 is low current control. When this bit is low (set to 0), the amount of current used in writing on the disk is reduced. The purpose of this is similar to write-precompensation: to reduce the interaction between data cells in the cylinders near the center of the disk. Low current reduces the intensity of the magnetic fields written on the disk. It is used starting with cylinder 128 (80H) for the ST506. Refer to the drive manual for other products.

Neglecting to use write-precompensation and low current control may result in the loss of reliability of data on the inner cylinders of the disk. For those writing their own driver software, use the drive manufacturer's specification to determine when to enable write-precompensation and low current.

Table 2-3: Format of the Select Head Byte

Bit #	Description
0	These two bits specify the drive number.
1	
2	These three bits specify the head number.
3	
4	
5	Not used.
6	Turns on low current when 0.
7	Turns on write-precompensation when 1.

2.1.7. DMA Address

The DMA (Disk Memory Access) address is the beginning location for data transfers. When reading or writing data to the disk, and also when formatting, these three bytes point to the first byte of the data buffer in main memory.

The DMA address is not maintained by the controller from one operation to the next. The controller never alters the DMA address in the command structure, so consecutive commands that refer to the same buffer and structure need not rewrite these bytes.

The controller asserts all 24 address bits of the IEEE 696/S-100 bus specification. In systems that do not decode the upper eight address lines, the extended page byte must still appear in the command structure, as the format of the structure is fixed at three bytes of information.

When data transfers cross 65K boundaries, the extended page address byte is incremented by 1. In systems not responding to the extended page address, this results in wrap-around, that is, the address after the highest 65K byte is Ø.

Table 2-4: Format of the DMA Address Bytes

Byte 4	Low byte of DMA address
Byte 5	High byte of DMA address
Byte 6	Extended address byte of DMA address

2.2. Operation Codes

The HDCDMA controller responds to seven operation codes. The codes provide access to all of the functions programmed into the controller. Each operation code uses some or all of the eight parameters that precede the code. The seven valid codes are as follows:

Table 2-5: Operation Codes

CODE	FUNCTION
Ø	Read data
1	Write data
2	Read header
3	Format track
4	Load constants
5	Sense status
6	No operation (NOP)

Read Data and Write Data will be discussed together. These are the most commonly used operations. Format Track is used in formatting disks, and is of special interests to those interested in designing systems with special timing or data organization constraints.

The Load Constants command sets up the step delay and head settle time, which are important in getting the maximum response from the new intelligent drives in terms of decreased seek latency. The Load Constants command also sets sector sizes, and must be used any time the controller is reset.

Read Header, Sense Status and No Operation codes are provided for the other essential control operations, and are discussed separately.

The status codes returned in the byte after the operation code are the same for all operations except Sense Status. When the operating system writes the command structure, it should store a Ø in the status byte. At the completion of a command, the channel transfers the status to this byte in the structure. The status codes are defined in the table that follows.

Table 2-6: Status Codes

Code	Description
ØØ	Busy
Ø1	Drive not ready
Ø4	Sector header not found
Ø5	Data not found (no data preamble)
Ø6	Data overrun (channel error)
Ø7	Data CRC error
Ø8	Write fault
Ø9	Sector header CRC error
AØ	Illegal command
FF	Successful completion

2.2.1. Read and Write Data

These are the basic HDCDMA controller commands. Their operation is almost identical. Each time the controller is started, it commences stepping, if requested, then transfers the parameters and the operation code. The controller then reads headers until it finds the header that matches the description in the four bytes of argument. If it fails to do so after reading 128 headers, it reports an error in the status byte, and halts.

Once the matching header is found and its CRC is correct, the controller requests the bus. The bus has about 2Ø microseconds to finish its current operation and respond to the controller. This is the time between checking the header CRC word and the detection of the mark that begins the data field. The controller will hold the bus until the entire sector has been transferred. Data transfers by the controller are in burst mode. This means that once the channel becomes the temporary bus master, it holds the bus until it has transferred one sector. The data rate of a ST5Ø6, for example, is one byte every 1.6 microseconds. This would be 1.64 milliseconds for a 1Ø24 byte sector to be transferred on the ST5Ø6.

The controller uses a three T-state bus cycle to transfer a byte of data. A fourth T-state may be required because the bus and the disk are running asynchronously. If the bus is unable to respond in time, a data overrun occurs and the read or write operation will terminate by reporting an overrun error in the status byte. The section on the DMA interface goes into more detail on the timing considerations.

In cases where the controller has reported an unsuccessful completion, such as CRC errors or overrun, the operating system should be prepared to retry the last command. This is done by pointing the Link Field to the beginning of the failed command, setting the number of steps to Ø, and repeating it until it is successful or the number of retries has been exhausted. Ten retries are suggested, although more or less can be used.

The Read and Write Data operations both use the previously described Select Head Byte and DMA address. The other variable that needs to be discussed in their use is the sector header. The four bytes of the argument list contain an image of the header to be matched. It is normally written as follows:

Table 2-7: Argument List for Header Information
(Recommended Format)

Byte#	Arg#	Description
7	Ø	Low byte of cylinder address
8	1	High byte of cylinder address
9	2	Head number
1Ø	3	Sector number

When the drives are initially formatted, it is possible to write ANY four bytes of data in the sector header. The purpose of this data is to verify that the correct cylinder and head is being accessed, and to identify sectors. If other sector header formats are desired, they may be written during formatting and successfully found during read header operations as long as the data in the argument list matches the written data in the sector header. This feature accommodates a wide variety of designs limited only by the four byte field and the designer's imagination.

The operation code for Read Data is Ø; the code for Write Data is 1. As an illustration, the following command structure will write the buffer located at ØØØØ8ØH to drive A, head number 2, cylinder number 64H, sector number 15. For this example, we will assume that some previous operation has left the heads positioned over cylinder 5AH and the previous link field contains ØØØ43H.

Table 2-8: Command Structure Example for Write Data Command

BYTE#	LOCATION	CONTENT	DESCRIPTION
Ø	43(hex)	ØØ(hex)	Select drive A, step in.
1	44	ØA	Low byte: step 1Ø steps.
2	45	ØØ	High byte: Ø * 256 steps.
3	46	48	Select drive A, head 2.
4	47	8Ø	Low byte of DMA address.
5	48	ØØ	High byte of DMA address.
6	49	ØØ	Extended address byte of DMA.
7	4A	64	Low byte of cylinder address.
8	4B	ØØ	High byte of cylinder address.
9	4C	Ø2	Head number.
1Ø	4D	ØF	Sector number 15.
11	4E	Ø1	Write Data Operation Code.
12	4F	ØØ	Space for status byte.
13	5Ø	43	Low byte of link address.
14	51	ØØ	High byte of link address.
15	52	ØØ	Extended address byte of link.

When a start command is given, the heads of drive A step in 1Ø cylinders. The controller next reads sector headers until a match is found, then data is written to the disk from memory starting at 8ØH. The sector size has been set by a Load Constants command, so the write operation continues until an entire sector has been written. The write-precompensation and low-current bits are turned off because the cylinder number is less than 8ØH.

Please note the address in the link address field. When this operation finishes, the next start command points the controller back to the beginning of the command structure. When used in this manner, the command structure resembles a table where the data for the desired operation is filled in, and the link field remains the same.

2.2.2. Format Track Operation

This operation is used in formatting disks. As in any formatting operation, any data written to the disk previously will be obliterated. Therefore, great care must be taken in the use of this command.

The Format Track operation formats a single track. This track is specified by the cylinder address and the head selected. During the operation, a four byte set of variable data is read for each sector from a buffer starting at the DMA address and written into the sector header.

To format an entire cylinder, each track is formatted by selecting a head with the Select Head Byte and providing header data.

Each sector header contains four bytes of variable data. The data that is to be written in the sector headers is located in a buffer pointed to by the DMA address. Four bytes of data that will **uniquely** define each sector of a particular drive must be written in the buffer before each track is formatted. A suggested format is given in the table that follows, but any four byte format may be used.

Table 2-9: Suggested Format for Sector Headers

Byte#	Description
Ø	Low byte of cylinder address
1	High byte of cylinder address
2	Head number
3	Sector number

EXAMPLE: The following is a memory dump of the data that would be used while formatting cylinder 41H and head 3 for 9 1Ø24 byte sectors, with the DMA field of the command structure set to 5ØØØH, and using the suggested format.

```
5ØØØ:   41ØØ Ø3 ØØ  41 ØØ Ø3 Ø1  41 ØØ Ø3 Ø2  41 ØØ Ø3 Ø3
5Ø1Ø:    41 ØØ Ø3 Ø4  41 ØØ Ø3 Ø5  41 ØØ Ø3 Ø6  41 ØØ Ø3 Ø7
5Ø2Ø:    41 ØØ Ø3 Ø8  41 ØØ Ø3 Ø9
```

The normal strategy for numbering sectors is to allow the operating system to map the physical sectors to logical sectors. The sectors on the disk are then numbered sequentially during formatting.

It is also possible to incorporate any desired sector interleave into the disk format simply by incrementing the sector number by the desired skew, modulo the number of sectors for track. The sector interleave is chosen to maximize certain aspects of system performance.

The Format Track command uses all four arguments of the command structure. The first argument is the intersector gap. This is the number of bytes of 4E's that are written between sectors. It is dependent upon sector size, with the larger sectors requiring bigger gaps.

The purpose of the intersector gaps is to allow for variations in the rotational velocity of the drives. For example, if the drive were rotating 1% faster while formatting, a sector written when the drive is moving slower will extend it over a longer section of the track. Suggested intersector gaps are given in the table in this section.

The intersector gap, also called Gap 3, is calculated by multiplying the total number of bytes per sector (this must include fixed overhead, which is 44 bytes per sector) times .Ø61, and taking the minimum of this value and 256. This provides a margin of error for 3% speed fluctuations.

Gap 3 = minimum of (((bytes/sector + 44) x .Ø61), 256)

The second argument is the complement of the number of sectors per track. Each time the Format Track command is executed, the controller begins writing, starting at the index for the number of sectors per track of header and data fields, followed by 4E's, until the index. For example, nine 1Ø24 byte sectors will fit on one track. Nine is ØØØØ 1ØØ1 in binary, so its complement is 1111 Ø11Ø or F6H.

The third argument is a code for the number of data bytes in each sector. Sector size codes are the complement of the codes used in the Load Constants Operation for sector sizes. The table that follows defines these codes for the five different sector sizes possible:

Table 2-1Ø: Sector Size Codes for Formatting

CODE	NUMBER OF BYTES
FF(hex)	128(decimal)
FE	256
FC	512
F8	1Ø24
FØ	2Ø48

The fourth argument is the data fill byte. While the controller is formatting a track, it is continually writing 4E's, zeroes for synchronization, the sector header, marks, CRC's or the data fill byte. The data field is written with the number of bytes per sector of the data fill byte. The data fill byte is E5 for CP/M *. The table that follows recaps the four arguments.

* C/PM is a trademark of Digital Research.

Table 2-11: Argument List for Format Track Operation: Code = 3

Byte#	Arg#	Description
7	Ø	Intersector Gap
8	1	Complement of number of sectors/track
9	2	Code for number of bytes/sector
A	3	Data fill byte

The table which follows lists the appropriate number of bytes for various sector sizes. It also provides information for choosing sector sizes.

Table 2-12: Sector Sizes

SECTOR SIZE:	128	256	512	1Ø24	2Ø48
Intersector Gap	1Ø	18	43	65	256
Whole sectors	56	32	17	9	4
Megabytes/disk*	4.38	5.Ø13	5.636	5.64Ø	5.Ø13
Intersector Gap	ØA(hex)	12	2B	41	FF
Complement #sector	C7(hex)	DF	EE	F6	FB
Sector size code	FF(hex)	FE	FC	F8	FØ
Data Fill Byte**	E5(hex)	E5	E5	E5	E5

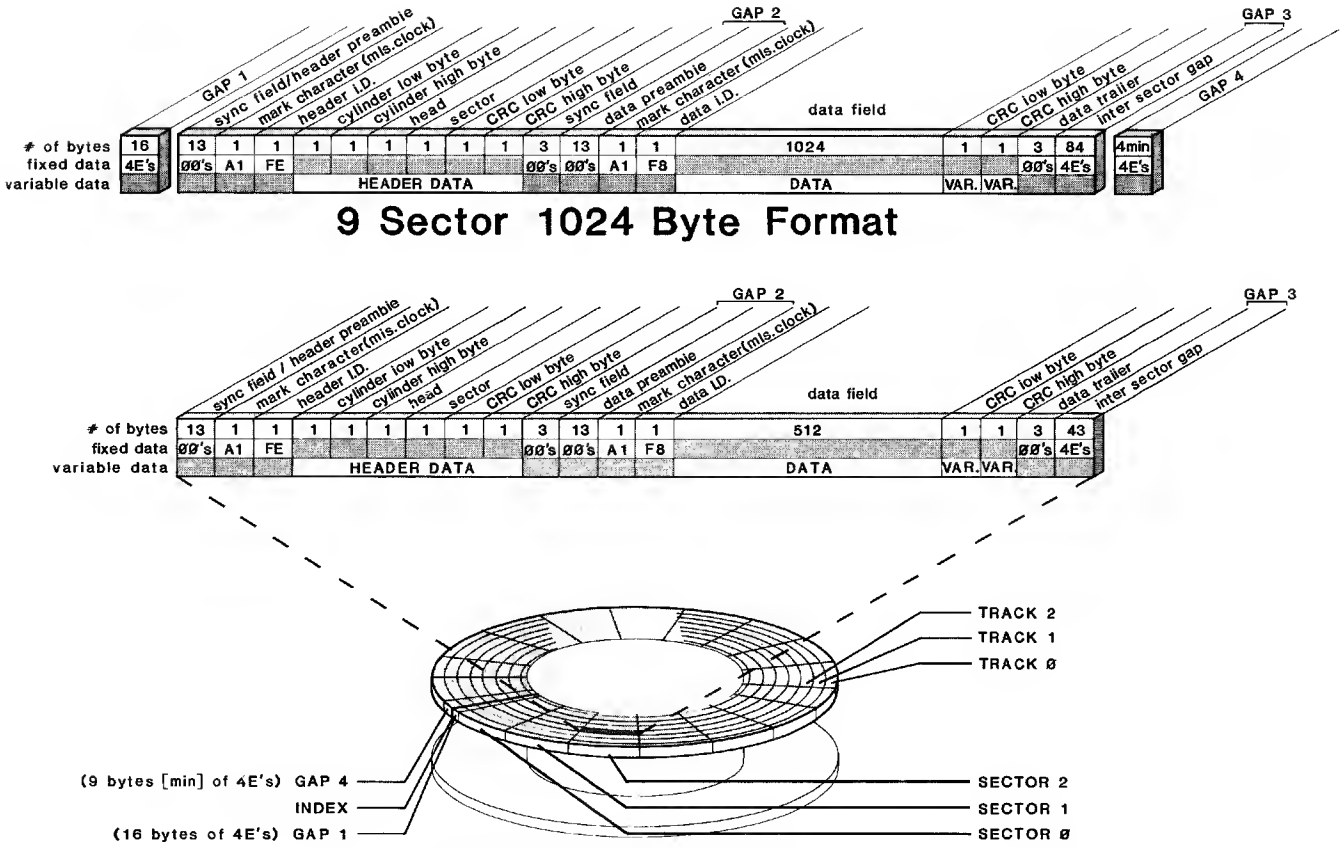
*These values are computed by multiplying number of sectors times the number of bytes per sector times the number of heads (4) times number of cylinders (153) for a ST5Ø6 hard disk. Not included is a fixed overhead of 44 bytes/sector, which includes sector header, 4 CRC bytes, 32 sync bytes (ØØ) and two preambles of 2 bytes each.

**The data fill byte is E5 for CP/M (a trademark of Digital Research).

The five sets of hex values in the preceding table contain the allowable values for all of the arguments used with theFormat Track Command. For example, 41, F6, F8 and E5 for 1K sectors.

The figure on the following page explicitly illustrates the format of data and headers as they actually appear on the disk. The 1Ø24 byte/sector format is that used by all Morrow Designs software, and maximizes data density and system performance. A second example is given of a different sector format for comparison.

Each track begins with Gap 1 followed by the required number of sectors, which are identical except for the sector header, and concludes with Gap 4. Gap 4 is used to fill the space between the last intersector gap (Gap 3) and the next index mark.



9 Sector 1024 Byte Format

17 Sector 512 Byte Format

Fig. 2-1: Byte Sectors - 1Ø24 and 512 Format

2.2.3. Read Headers Command

This command is used to read the first eight bytes that occur after a preamble. This may be either a sector header, including the mark, the FE, two bytes of cylinder address, head, sector number and 2 CRC bytes or the mark, F8 and 6 bytes at the beginning of a sector of data. These eight bytes are transferred to the buffer pointed to by the DMA address.

To distinguish between a sector ID field and data, the second byte will be an FE when a sector header is read and an F8 when reading a data field. When a data field is read, a CRC error will also be reported.

If a data field was read, retrying the Read Header command will usually find the next sector header. The operation code for Read Header is 2. The argument field is not used in this operation.

2.2.4. Load Constants Operation

The Load Constants Operation is required to initialize the controller. The HDCDMA allows software control of the two variables that affect seek times, step delay and head settle time, and the sector size. The sector size is used to inform the controller of the number of bytes to be transferred in a Read or Write Data operation. There are no default values for these variables, so this is the initialization operation after any reset or before using different types of drives.

A drive is considered to be a different type if it requires a different step delay or head settle time, or different sector size. When any of these three values change, Load Constants must be used before selecting the drive.

The new breed of 5 1/4" and 8" hard disks may have on-board intelligence for fast seeks. Normally, the controller sends a step signal, waits the step delay time and sends the next step signal until stepping is completed. In the newer drives, step signals are accepted in a stream as fast as the controller can send them. The drive will then ramp up its stepper and ramp down again to provide the fastest possible seek time.

At the end of this ramp up and ramp down, the drive will make true the SEEK COMPLETE* signal. The HDCDMA controller monitors this status line, and begins the head settle time after SEEK COMPLETE* goes true. In some drives, SEEK COMPLETE* will not be true until after the heads have settled. In this case, a head settle time of Ø may be used.

* The asterisk implies negative logic. When SEEK COMPLETE* is Ø it is true.

In drives with built-in ramping, a Ø seek delay time should be used. If the drive waits until after the heads have settled to make SEEK COMPLETE* true, then a head settle time of Ø should be used. In drives that do not have either of these features, the manufacturer's recommended delay and settle times should be used.

The step delay times may vary between 11 microseconds and 12.8 milliseconds in increments of 1ØØ microseconds. The controller takes 11 microseconds to issue each step pulse, so a 2ØØ microsecond delay actually lasts 211 microseconds. The head settle times may be in the range of Ø to 25.6 milliseconds in increments of 1ØØ microseconds.

The Load Constants Operation has a further function. By setting bit 7 of the step delay time to a 1, an interrupt signal will be generated at the completion of all controller operations. This signal is reset by issuing a start command to the HDCDMA, and is generated at the completion of all commands until a Load Constants Command resets bit 7.

The interrupt signal is brought to a strappable pad on the lower left of the board, and must be connected to the PINT/ or one of the vectored interrupt lines before it can reach the bus. Please refer to the section on Configuring Features for more information.

Table 2-13: Arguments for Load Constants Command

Byte#	Arg#	Description
7	Ø	Not used
8	1	Step delay time and interrupt enable
9	2	Head settle time
A	3	Sector size code

Table 2-14: Argument 1 Format: Step Delay and Interrupt Bit

Bit#:	7	6	5	4	3	2	1	Ø
Use :	Set/reset for Interrupts.	Choose for Ø to 128 1ØØ microsecond step delay intervals.						

Table 2-15: Sector Size Codes for Load Constants Command

Number of Bytes	Code
128	Ø
256	1
512	3
1Ø24	7
2Ø48	F

2.2.5. Sense Status Operation

The Sense Status Operation returns status information from the selected drive. Operation completion is indicated by a non-zero value in the status byte. The meaning of this byte is explained below. In this case, the Status Byte will not correspond to any of the Status Codes described earlier.

Most of the information returned in the Status byte is used by the firmware in the controller and is not significant to normal use. One bit in particular, however, should be checked whenever a drive is selected.

The DRIVE READY* signal is generated by a drive whenever it is ready and the speed of the disk(s) has stabilized. Attempting to access the drive for a read or write operation before it is ready may result in locking up the controller. This is because it will be unable to synchronize with an erratically moving disk. The controller would have to be reset in this case.

It is suggested that the DRIVE READY* line be polled every time a new drive is selected. This will prevent the controller from being hung up trying to synchronize with a not-ready drive.

Four other bits of information are provided by the Sense Status Operation. Bit Ø represents the track Ø detect. Whenever the heads are over the outermost cylinder, TRACK ZERO* will be true, that is, Ø.

The WRITE FAULT* status is a signal from the drive that reports illegal commands or conditions. Examples of these conditions would be low voltage for a write operation, or trying to read and write simultaneously. The manufacturer's specification for the drive should be referred to for more exact meaning of this bit. Software may want to check this bit at the end of all Write operations.

The SEEK COMPLETE* has been mentioned in connection with the Load Constants Operation. It is made to be true either after a step, stepping or head settling is complete. Again, please refer to the drive manufacturer's specifications for the exact timing of this signal.

The NRZ INDEX is connected through a divider to the index detector of the selected drive. Each time the index is detected, the NRZ INDEX changes state. That is, during one revolution it is a 1, and on the next revolution, a Ø.

* As mentioned previously, * is used to indicate negative logic.

The Sense Status operation returns the information in the Status Byte, which follows the operation code. This code should be initialized to a Ø in the command structure. When the operation is complete, the Status Byte will be non-zero, and the lower 5 bits will represent the status of the selected drive. No arguments are required.

Table 2-16: Status Byte after Sense Status Operation

Bit#	Meaning
Ø	TRACK ZERO* detect
1	WRITE FAULT* signal
2	DRIVE READY* signal
3	SEEK COMPLETED* status
4	NRZ INDEX, alternates with each revolution
5	
6	These bits set to 1 after Sense Status
7	

2.2.6. No Operation Command

This command may be used for stepping the heads without performing data transfers, or as a means of setting the link field to point to the next command structure to be executed. The operation code for this command is 6.

This is the command that would be used to perform a recalibration. To recalibrate a drive, the SELDRV byte (Byte Ø) is used to select the drive and the stepping-out direction. In the next two bytes, a value for the number of steps greater than the maximum number of tracks for that drive is used. When the heads reach track Ø, the controller stops seeking.

Fast seeking must not be used while recalibrating. A Load Constants command is used to set the step delay to the appropriate interval for slow stepping. A second Load Constants command is used to restore the drive to fast seeking. If fast seeking were used, the heads might reach track Ø while the seek speed is being ramped up and cause damage to the drive.

3. DMA INTERFACE

The HDCDMA Controller uses two types of Direct Memory Access: cycle stealing and burst mode. In cycle stealing, the channel requests the bus to transfer one or several bytes. If the controller loses control of the bus during the transfer, it detects an overrun and will try again.

The time spent waiting for control of the bus is not critical during cycle stealing because there is no real-time activity ongoing. The HDCDMA uses cycle stealing for reading the command structure and reporting status.

During disk transfers, burst mode is used. Burst mode is sometimes referred to as "hogging". During burst mode, the channel does not release the bus until it has finished the entire data transfer. Also, interrupts and other channels requiring use of the bus are inhibited. This should be taken into account if other devices with specific timing constraints are used on the bus.

Burstmode is used to accommodate the rapid data transfer rate of hard disks. Data is transmitted at the rate of 5 million bits*second with ST5Ø6 drives. This translates to a byte of data ready every 1.6 microseconds.

The channel requests the bus for burst mode when it successfully matches the header field of the target sector. This allows about 2Ø microseconds for bus acquisition and should be sufficient for the CPU and any other non-burst DMA devices to finish their activities.

In systems requiring wait states, the HDCDMA will respond to the bus RDY lines. The net speed (a combination of the bus clock speed and wait states) must be able to keep pace with the asynchronous data transfer rate of the disk.

The controller requires three bus T-states to transfer one byte of disk data. Because the bus clock and the data rate of the disk are asynchronous, an additional 1 bus T-state may be required to synchronize the transfer of a byte of disk data. To calculate the minimum bus clock, multiply the disk rate (in bytes per second) times 4 plus the number of wait states. In a system without wait states and using a ST5Ø6, this would be 2.5 megahertz.

$$\text{Minimum Bus Clock} = \text{Disk Data Rate (Bytes/Second)} \times (\ 4\ \text{(T-states)} + \text{number of wait states}\ )$$

Greatest system performance will be obtained if floppy (DJDMA) and hard disk DMA controllers are allowed to operate concurrently. If this is done, there will occasionally be collisions, that is, situations where both controllers require the bus at the same time.

When this occurs, the floppy disk controller will probably be the one to experience overrun, since the hard disk controller "hogs" the bus. System software should be prepared to repeat either operation in the event of an overrun.

3.1. Temporary Master

The controller acts as a TMA, or temporary master. The CPU is the permanent bus master. When a temporary master wants control of the bus to perform a DMA operation, it requests the bus by asserting a signal called HOLD*. The CPU, as permanent master, monitors the HOLD* signal. When the CPU finishes the current bus cycle, it acknowledges HOLD* by asserting the signal called pHLDA (processor hold acknowledge).

The temporary master takes control over from the CPU in a two step process. First, the CPU's data-out, address and status lines are disabled using DODSB*, ADSB* and SDSB*, and enabling the temporary master's control drivers. During this time, both the permanent master and the temporary master are driving control lines.

Second, CDSB* (control disable) is made true, giving the temporary master complete control of the bus. The temporary master has now taken the place of the CPU on the bus. Returning control to the CPU (permanent master) is the same process in reverse ending when HOLD* is made false.

So far, the process has been described as if only one temporary master wanted control of the bus. There can be up to 16 temporary masters on the bus. When there are more than one temporary master, they use the four DMA lines to decide who gets to assert HOLD*.

Any device requesting the bus places its TMA priority level on the bus if HOLD* is not already true, and circuitry on the device decides if it has the highest priority. The device with the highest priority (ØF hex is highest) asserts HOLD* and leaves its priority on the DMA lines until it removes HOLD* from the bus. This normally results in the first TMA to request the bus getting the bus. For a precise definition of DMA arbitration, S-1ØØ/IEEE696 specification should be considered the last word. The design of the HDCDMA follows the specification.

4. CONFIGURATION FEATURES

There are two features of the HDCDMA Controller that are set by changing slide-on jumpers or adding wire jumpers. These are: the DMA priority level and the interrupt line asserted.

4.1. DMA Priority Level

The DMA priority level is set using slide on jumpers. There are four sets of two pins each, labeled J1 - J4. These are located in the lower left hand of the component side of the board.

J1 corresponds to DMA-Ø, J2 to DMA-1, J3 to DMA-2 and J4 to DMA-3. When a pair of pins are not jumpered together, they represent a high, or 1. Jumpering a pair of pins together sets that DMA line to a zero. By selectively connecting or leaving unconnected the four pairs of jumpers, any priority between Ø and 15 can be selected.

The highest DMA priority is 15. For the HDCDMA, this means removing all four slide-on jumpers. This is the suggested priority for the HDCDMA. To select a priority of 6, jumpers would be used to connect J1 and J4. The following table summarizes these jumpers:

Table 4-1: DMA Priority Jumpers: J1 - J4

Pair	DMA Assignment	
J1	DMA-Ø	Least significant bit
J2	DMA-1	
J3	DMA-2	
J4	DMA-3	Most significant bit

4.2. HDCDMA Interrupts

The output of the controller's interrupt request buffer is brought to a pad labled J5. This is in the same area as the DMA priority jumpers. Located between this pad and the S-1ØØ bus connection are 9 pads that J5 may be connected to.

PINT* is a direct line to the interrupt input of the CPU board. This line is normally used by a device, such as a MULT/IO board or a programmable interrupt controller, which prioritizes interrupt requests, and provides other interrupt support. The HDCDMA does not have the necessary hardware to provide interrupt support, such as gating the address of a service routine onto the bus at interrupt acknowledge.

Therefore, J5 will normally be connected to one of the eight vectored interrupt lines. The vectored interrupt lines must be connected to interrupt controller circuitry before reaching the main CPU. They are organized so that VIØ* has the highest priority, and VI7* has the lowest.

Interrupt request generated by the HDCDMA signifies the completion of its last command operation. At this point, the status should be checked. If the operation was successful, tasks that are waiting for the data or the controller may proceed. Otherwise, the last operation should be retried.

The interrupt line of the HDCDMA is cleared by the start of the next operation. The completion of every operation sets the interrupt line until interrupts are disabled. Bit 7 of the Step Delay Byte is used to enable/disable interrupts. Please refer to the Load Constants Command.

4.3. Connecting Cables

There are five male connectors located along the top edge of the HDCDMA: one 34/50 pin connector and four 20 pin connectors. These are labeled P1 - P5 starting on the right side of the board.

The 34/50 pin connector, labeled P1, carries the control and status information to all drives that are connected to the controller. The 34 pin connectors are used with 5 1/4" drives, and 50 pin connectors with 8" drives.

The drives are daisy-chained along this cable, and their order on this cable is not important. For example, the fourth drive could be connected first, the third second and so on.

The other four connectors must be attached to a particular drive. P2 should be connected to the drive jumpered as 1, P3 to drive 2, P4 to drive 3 and P5 to drive 4. When the controller selects a drive, only one of these four connectors is enabled. If a drive has been configured to respond as drive 1, it MUST be connected to P2.

The 20 pin cables carry read and write data between the controller and a particular drive. A clock provides a timing signal that is 1/16 the bit rate, which is used by some of the older hard disk designs.

When the cables are connected to the HDCDMA, they should extend over the back (solder side) of the board. At the drive end, the 20 conductor cables should pass over the center of the back of the drive cabinet. For cabinets with the connectors at the top, the cable will extend down from the connector.

The control/status cable (34/50 conductors) should pass over the center of the back of each drive cabinet, the same as the data cables. When there are several drives connected, the control/status cable that leads to the next drive should not extend over the back of the drive cabinet.

These descriptions of cable connections will work with current Morrow Designs products. For other configurations, here are some points to keep in mind:

1) On each connector on the HDCDMA, pin 1 is on the right. Therefore, the other end of the cable should be attached to the drive so that the conductor in the cable corresponding to pin 1 is connected to pin 1 on the drive PC.

2) Once the control/status cable is properly connected, the drive will respond to select and step commands. If the drive steps, but is unable to read or write data, the data cable may be connected to the wrong HDCDMA connector (P2 - P5), or the cable may need to be reversed.

3) The FORMAT program will select a drive and send stepping pulses while attempting to format the disk. If the formatting part of the program completes, and the verify fails (produces a LONG list of errors), the data cable connection should be examined.

4.3.1. Cable Pinout

As explained in the previous section, there are two types of cables used with the HDCDMA: the data cables and the control/status cable. The data cables form a direct connection between each drive and the controller. They carry only three signals in the form of current-loop pairs: MFM (modified frequency modulated) Read Data, MFM Write Data and a Timing Clock.

The Timing Clock produces a signal that is 1/16 the bit rate of the controller. This is provided for drives which require this signal.

Connectors P2 - P5 have identical pinouts. The description of these follows:

Table 4-2: Data Cable Pinouts

	MFM		MFM		TIME				
G	RD+	G	WD+	G	CLK+	NC	NC	NC	NC
19	17	15	13	11	9	7	5	3	1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
20	18	16	14	12	10	8	6	4	2
G	MFM	G	MFM	G	TIME	G	G	G	G
	RD-		WD-		CLK-				

The control/status cable is daisy-chained between the controller and all drives. The pinouts of the 34 pin cable are identical with the 50 pin (for 8" drives) for the first 34 pins. Some of the signals used in the first 34 pins are duplicated in the last 16 of the 50 pin cables. A diagram of a 34 pin connector appears below. The pins that are repeated in the 50 pin connector appear in brackets. The top of the drawing is located at the upper right-hand corner of the HDCDMA board and is labeled P1.

Table 4-3: 34 Pin Connector

LOW	2 .	. 1	G
HS4/	4 .	. 3	G
WRITE GATE/[40]	6 .	. 5	G
SEEK DONE/	8 .	. 7	G
TRACK 0/ [42]	10 .	. 9	G
WR FAULT/[44]	12 .	. 11	G
HS0/	14 .	. 13	G
NC	16 .	. 15	G
HS2/	18 .	. 17	G
INDEX	20 .	. 19	G
DRIVE READY/	22 .	. 21	G
STEP/ [36]	24 .	. 23	G
DS1/	26 .	. 25	G
DS2/	28 .	. 27	G
DS3/	30 .	. 29	G
DS4/	32 .	. 31	G
DIRECTION/	34 .	. 33	G

G - ground
DSn/ - Drive Select n
HSm/ - Head Select m (multiplexed 1 of 8)

5. BOOTSTRAPPING

The HDCDMA may be bootstrapped by executing a short program external to the controller. This program may reside either on a floppy disk (for example, BOOTMW for C/PM*), or in PROM.

The Decision I PROM has a switch selected program for booting the HDCDMA. When the system is powered up or reset, this program is executed. Please refer to the Decision I user's manual for instructions for setting switches.

A brief outline of a program to boot the HDCDMA follows:

A Sense Status Command is executed until the DRIVE READY/ line becomes true (bit 2 = Ø);

A Load Constants Command sets the Step Delay and Head settle time;

A Read Data command is started that Homes the heads of the selected drive and reads sector 1 into main memory; the status byte should be monitored for successful completion, and retries issued if necessary;

When the Read is complete, the CPU begins executing code that was read, starting with the first byte.

Before the HDCDMA may be booted or used, it must be formatted, and the system software copied to it. The FORMATMW, MOVCPM and SYSGEN programs and appropriate documentation are provided with C/PM* systems.

* C/PM is a trademark of Digital Research.

COMMAND STRUCTURE FORMAT

BYTE	NAME	FUNCTION
Ø	SELDRV	Select drive and step direction
1	STEP-L	Low byte of number of steps
2	STEP-H	High byte of number of steps
3	SEL-HD	Select drive, head, write pre-compensation and low-current
4	DMA-L	Low byte of DMA address
5	DMA-H	High byte of DMA address
6	DMA-E	Extended address byte of DMA
7	ARG-Ø	Parameter list for Operation codes
8	ARG-1	
9	ARG-2	
A	ARG-3	
B	OPCODE	Operation code
C	STATUS	Completion status is stored here
D	NEXT-L	Low byte of link address
E	NEXT-H	High byte of link address
F	NEXT-E	Extended address byte of link address

Format of BYTE Ø: DRVSEL

Bit #	7	6	5	4	3	2	1	Ø
BYTE Ø:	Ø	Ø	Ø	In/Out	Ø	Ø	Drl	DrØ

Bits Ø and 1 (DrØ and Drl) are used to select drives with a binary number between Ø and 3.

Bit 4 is used to select stepping DIRECTION: if it equals Ø, step in; if bit 4 = 1, step out.

FORMAT OF THE SELECT HEAD BYTE: BYTE 3

Bit #	Description
Ø	These two bits specify the drive number.
1	
2	These three bits specify the head number.
3	
4	
5	Not used.
6	Turns on low current when Ø.
7	Turns on write-precompensation when 1.

OPERATION CODES: BYTE B

CODE	FUNCTION
Ø	Read data
1	Write data
2	Read header
3	Format track
4	Load constants
5	Sense status
6	No operation (NOP)

STATUS CODES: BYTE C

CODE	DESCRIPTION
ØØ	Busy
Ø1	Drive not ready
Ø4	Sector header not found
Ø5	Data not found (no data preamble)
Ø6	Data overrun (channel error)
Ø7	Data CRC error
Ø8	Write fault
Ø9	Sector header CRC error
AØ	Illegal command
FF	Successful completion

NOTE: These codes are returned in the Status Byte when executing any Opcode except Sense Status.

ARGUMENT LIST FOR READ/WRITE OPERATIONS
(Recommended Format)

Byte#	Arg#	Description
7	Ø	Low byte of cylinder address
8	1	High byte of cylinder address
9	2	Head number
A	3	Sector number

ARGUMENT LIST FOR FORMAT TRACK COMMAND

Byte#	Arg#	Description
7	Ø	Intersector Gap
8	1	Complement of number of sectors/track
9	2	Code for number of bytes/sector
A	3	Data fill byte

The table which follows lists the appropriate number of bytes for various sector sizes. It also provides information for choosing sector sizes and the legal arguments for the Format Track Command.

SECTOR SIZES

SECTOR SIZE	128	256	512	1Ø24	2Ø48
Intersector Gap	1Ø	18	43	65	256
Whole sectors	56	32	17	9	4
Megabytes/disk	4.38	5.Ø13	5.636	5.64Ø	5.Ø13
Intersector Gap	ØA(hex)	12	2B	41	FF
Complement #sector	C7(hex)	DF	EE	F6	FB
Sector size code	FF(hex)	FE	FC	F8	FØ
Data Fill Byte	E5(hex)	E5	E5	E5	E5

ARGUMENTS FOR LOAD CONSTANTS COMMAND

Byte#	Arg#	Description
7	Ø	Not used
8	1	Step delay time and interrupt enable
9	2	Head settle time
A	3	Sector size code

STEP DELAY AND INTERRUPT BIT: BYTE 8 OF LOAD CONSTANTS

Bit#	7	6	5	4	3	2	1	Ø
Use	Set/reset for Interrupts.	Choose for Ø to 128 step delay intervals.				1ØØ microsecond		

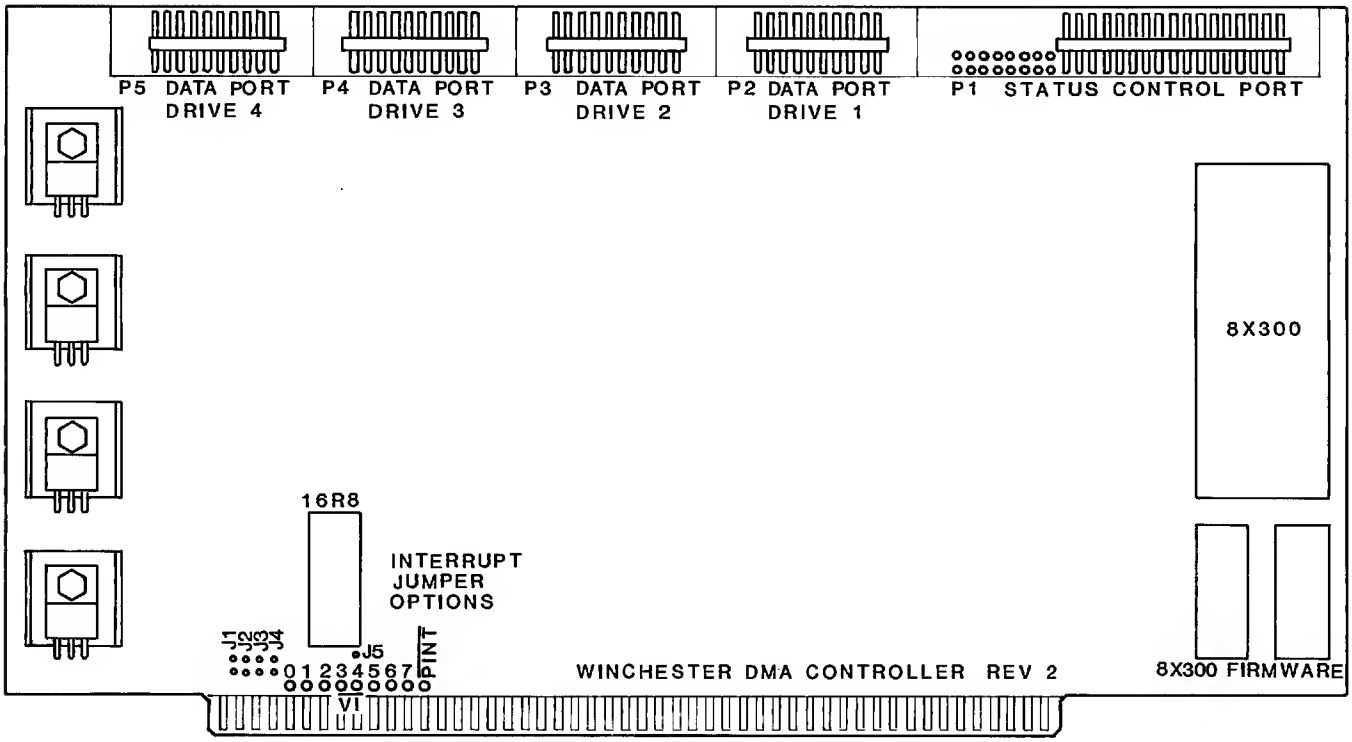
SECTOR SIZE CODES FOR LOAD CONSTANTS COMMAND

Number of Bytes	Code
128	Ø
256	1
512	3
1Ø24	7
2Ø48	F

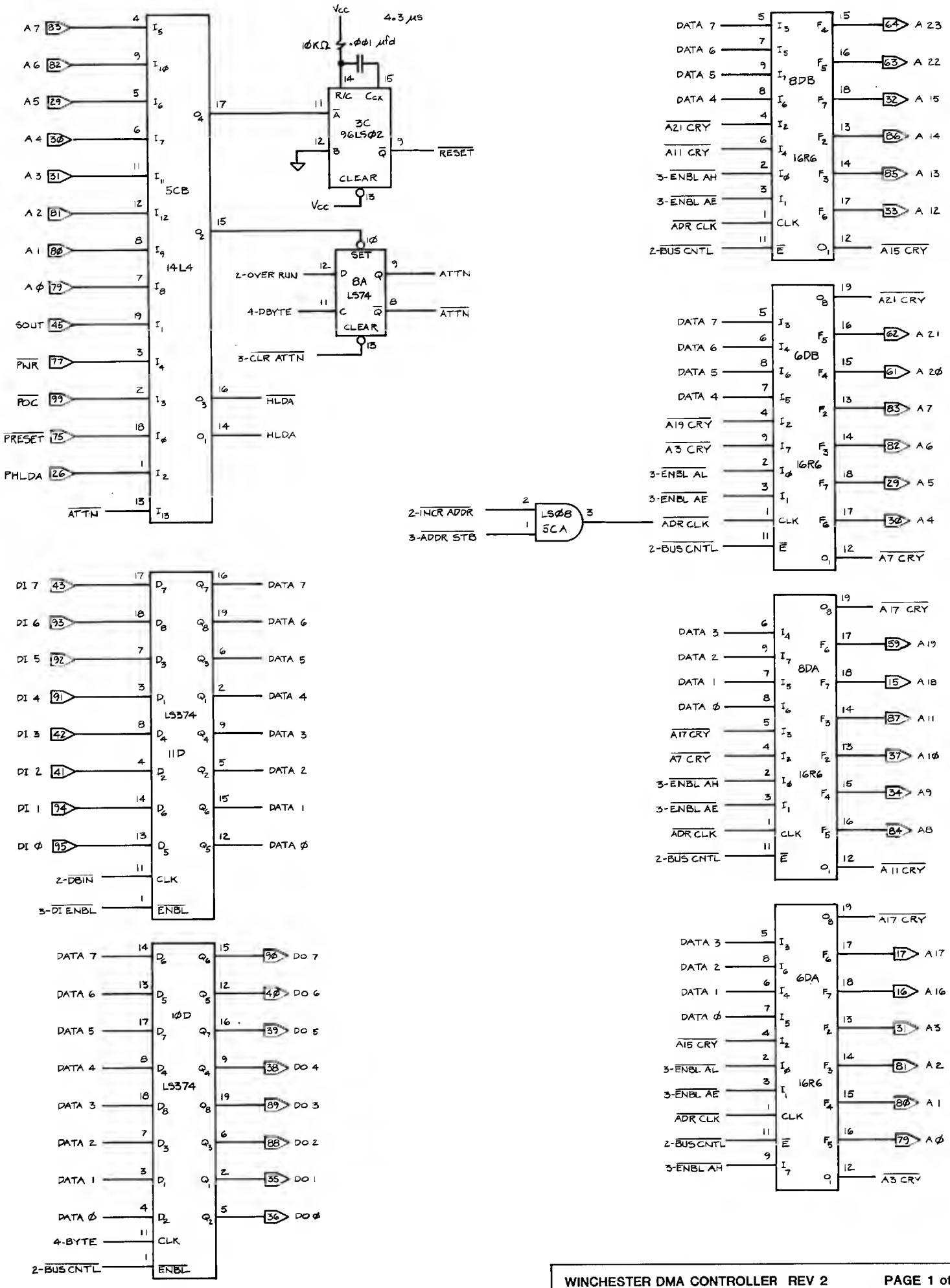
STATUS BYTE AFTER SENSE STATUS OPERATION

Bit#	Meaning
Ø	TRACK ZERO* detect
1	WRITE FAULT* signal
2	DRIVE READY* signal
3	SEEK COMPLETED* status
4	NRZ INDEX, alternates with each revolution
5	
6	These bits set to 1 after Sense Status
7	

COMPONENT LAYOUT/SCHEMATIC

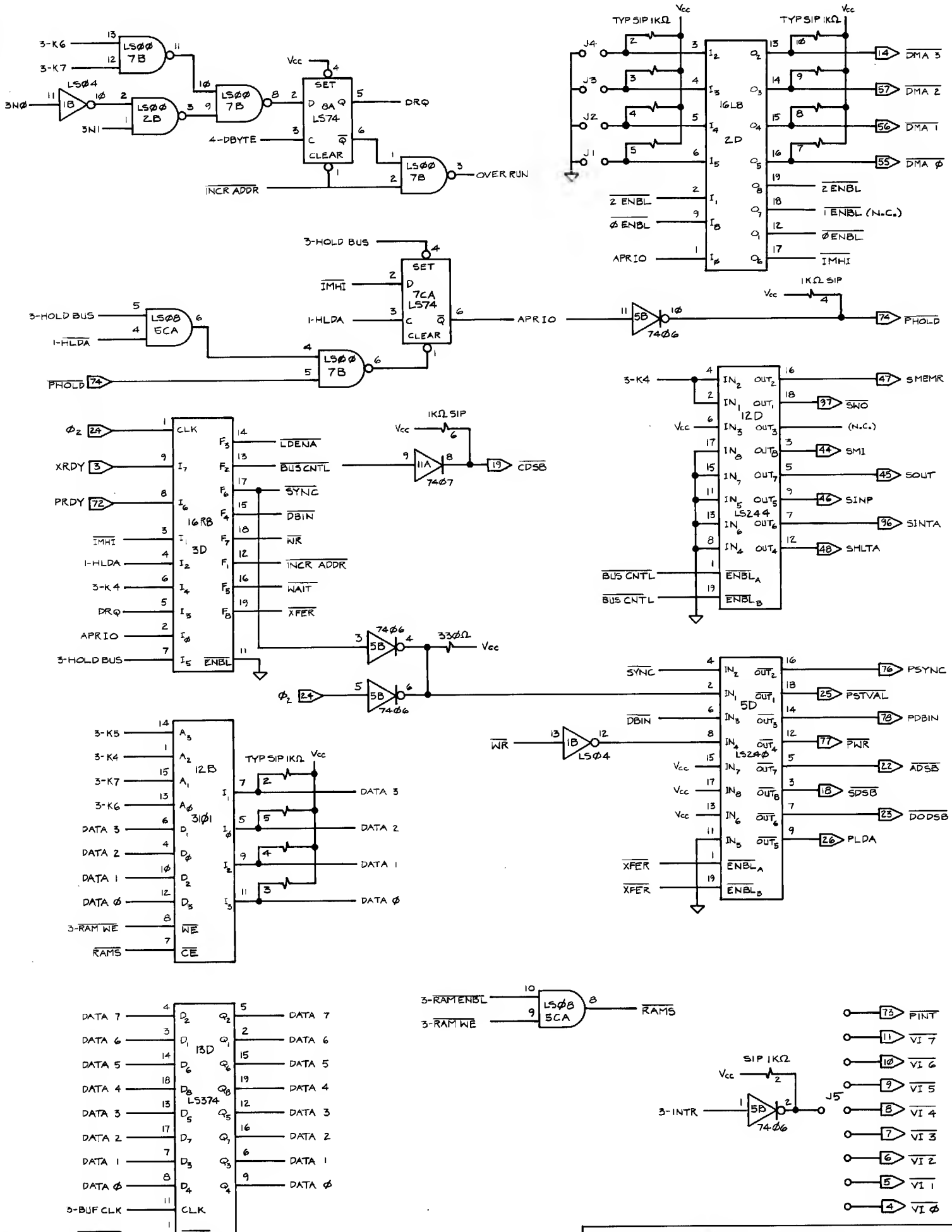


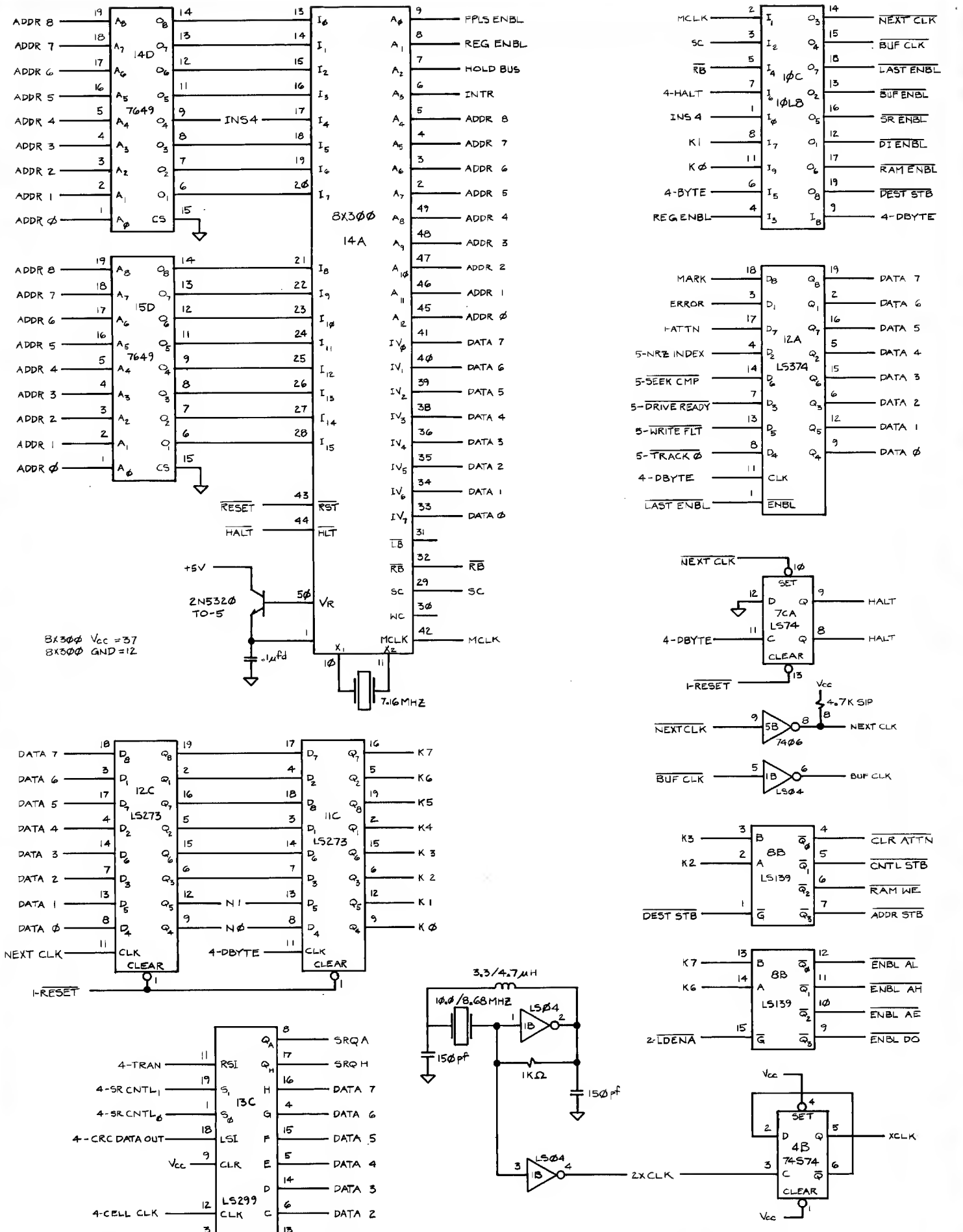
HD/DMA Component Layout



WINCHESTER DMA CONTROLLER REV 2 PAGE 1 of 5

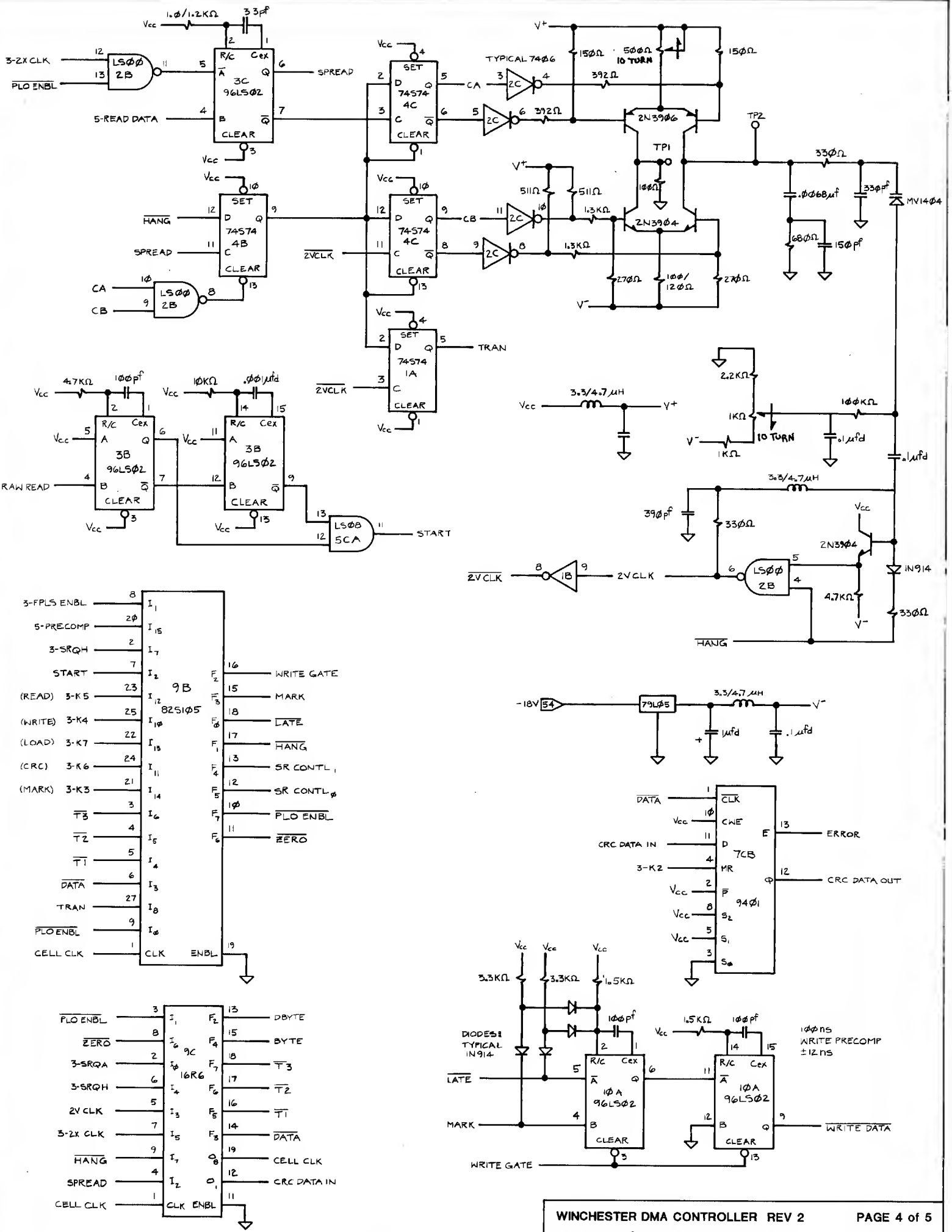
SELECT LOGIC & DATA BUFFERS





WINCHESTER DMA CONTROLLER REV 2

DATA TRANSFER LOGIC



WINCHESTER DMA CONTROLLER REV 2

DATA SEPARATION & ENCODING LOGIC

PAGE 4 of 5